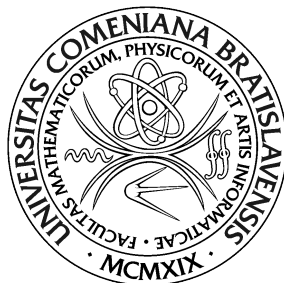


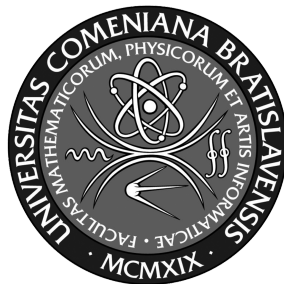
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



VIZUÁLNY SYSTÉM PRE
INTERAKCIU ĽUDSKÉHO
UČITEĽA S HUMANOIDNÝM
ROBOTOM

Diplomová práca

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



VIZUÁLNY SYSTÉM PRE INTERAKCIU ĽUDSKÉHO UČITEĽA S HUMANOIDNÝM ROBOTOM

Diplomová práca

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: Ing. Viktor Kocur, PhD.

Bratislava, 2022

Bc. Nicolas Orság



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Nicolas Orság
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vizuálny systém pre interakciu ľudského učiteľa s humanoidným robotom
Visual system for interaction of a human teacher with a humanoid robot

Anotácia: Toto zadanie je súčasťou projektu interakcie ľudského učiteľa s robotom. Robot pri tejto interakcii manipuluje jednoduchými objektmi na základe pokynov od ľudského učiteľa. Pre tento účel je tak vhodné aby robot dokázal správne detegovať pozíciu jednoduchých, objektov, učiteľových a svoje ruky. Následne je potrebné aby robot dokázal rozpoznať zadané inštrukcie.

Cieľ: Cieľom tejto práce je navrhnuť, implementovať a otestovať systém ktorý na základe vstupných stereo dát z kamier v robotovi NICO deteguje pozíciu jednoduchých objektov, učiteľovej ruky a robotových rúk a následne rozpozná gestá od učiteľa. Súčasťou práce bude prehľad existujúcich riešení detekcie objektov v stereo snímkach a rozpoznávaní gest učiteľa. Navrhnutý systém bude vyhodnotený v kontexte prebiehajúceho projektu interakcie ľudského učiteľa s robotom.

Vedúci: Ing. Viktor Kocur, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.
Dátum zadania: 05.10.2021

Dátum schválenia: 06.10.2021
prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som
vypracoval samostatne len s použitím uvedenej literatúry
a za pomoci konzultácií u môjho školiteľa.

Bratislava, 2022

.....

Bc. Nicolas Orság

Pod'akovanie

Abstrakt

Oblasť interakcie človeka a robota zaznamenala v posledných rokoch výrazný pokrok, pričom humanoidné roboty sa vyvíjajú pre široké spektrum aplikácií. Efektívna interakcia medzi týmito robotmi a ich používateľmi je však stále výzvou. Táto práca predstavuje vizuálny systém interakcie medzi ľudským učiteľom a humanoidným robotom. Systém využíva few-shot object detection algoritmy na učenie sa a rozpoznávanie nových objektov a gest.

Kľúčové slová: Počítačové videnie, Humanoidný robot, Hlboké učenie, Konvolučné neurónové siete, Few-shot object detection

Abstract

The field of human-robot interaction has seen significant progress in recent years, with humanoid robots being developed for a wide range of applications. However, effective interaction between these robots and their human users is still a challenge. This thesis presents a visual system for interaction between a human teacher and a humanoid robot. The system uses few-shot object detection algorithms for learning and recognising new objects and gestures.

Keywords: Computer vision, Humanoid robot, Deep learning, Convolutional neural networks, Few-shot object detection

Obsah

1	Úvod	1
2	Motivácia	2
3	Úvod do problematiky	3
3.1	Počítačové videnie	3
3.2	Príznaky	4
3.3	Objektová detekcia	5
3.3.1	Tradičné metódy	5
3.3.2	Metódy hlbokého učenia	6
3.4	Few-shot object detection(FSOD)	15
3.4.1	Prístupy k FSOD	16
3.4.2	Datasety pre vyhodnotenie FSOD	17
3.4.3	Aktuálne riešenia FSOD	17
4	Predchádzajúce riešenia	21
5	Návrh modelu	22
6	Implementácia	23
7	Výskum a výsledky	24

<i>OBSAH</i>	ix
7.1 Testovanie Frustratingly simple few-shot object detection . .	26
8 Záver	29

Kapitola 1

Úvod

Interakcia medzi ľuďmi a robotmi sa stáva čoraz dôležitejšou oblasťou v posledných rokoch vďaka rozvoju robotiky a jej aplikácií v rôznych odvetviach. Avšak účinná interakcia medzi ľuďmi a robotmi stále predstavuje výzvu, najmä v úlohách, ktoré zahŕňajú prirodzenú komunikáciu a ľudské gestá.

Cieľom tejto práce je vyvinúť vizuálny systém, ktorý umožní humano-idným robotom pochopiť a reagovať na základné gestá ľudského učiteľa a učenie sa rozpoznávať nové objekty.

Táto práca sa bude snažiť prispieť k oblasti interakcie medzi ľuďmi a robotmi vyvinutím vizuálneho systému na rozpoznávanie gest a objektov, pomocou preskúmania a použitia aktuálnych prístupov few-shot object detection, ktorý by mohol zlepšiť interakciu medzi ľudskými učiteľmi a humanoidnými robotmi.

Kapitola 2

Motivácia

V posledných rokoch je rastúci záujem o používanie robotov v rôznych odvetviach ako napríklad školstvo, zdravotníctvo, zábava a priemyselná výroba. Avšak na to aby roboti vedeli plniť úlohy efektívne, musia byť schopný prirodzene interagovať s ľuďmi. Toto je náročné veľmi náročné pri humanoidných robotoch, ktorý by mali napodobovať ľudský vzhľad a správanie.

Cieľom tejto diplomovej práce je vytvorenie vizuálneho systému, ktorý umožní humanoidnému robotovi rozpoznať učiteľové gestá a objekty pomocou few-shot object detection algoritmov, ktoré sú založené na konvolučných neuronových sieťach a hlbokom učení.

Kapitola 3

Úvod do problematiky

V tejto úvodnej kapitole si popíšeme a vysvetlíme základné pojmy, ktorých znalosť je nevyhnutná v našej práci. Vysvetlíme si ako funguje počítačové videnie, objektová detekcia, konvolučné neuronové siete a taktiež sa zameriame na výskum objektovej detekcie pri malej trénovacej množine (Few shot object detection), ktorý budeme chcieť využiť v našej práci pre učenie nových objektov aj z veľmi malého množstva dát.

3.1 Počítačové videnie

Počítačové videnie je súčasnej dobe veľmi rastúci a progresívny smer v informatike. Snaží sa priblížiť vnímaniu sveta z pohľadu ľudského oka, ktoré je pre nás prirodzené a automaticky sme schopný rozpoznávať objekty, farby a kontext toho čo vidíme. Avšak plné sémanticke pochopenie videnej reality je veľmi komplexné a zatiaľ nie sme schopný ho získať spracovaním digitálneho obrazu. Hlavne preto, že pochopenie obrazu môže vyplývať zo súvislostí, ktoré nie sú súčasťou obrazu.

Avšak počítačové videnie sa posúva veľmi rýchlo vpred. Neustále vyni-

kajú nové algoritmy a prístupy či už na detekciu objektov alebo klasifikáciu obrazu. Medzi základné problémy počítačového videnia patrí klasifikácia, objektová detekcia a segmentácia. Pri klasifikácii sa snažíme obraz priradiť do jednej z tried. V objektovej detekcii sa snažíme v obraze určiť oblasti všetkých známych objektov a priradiť ich do tried. A pri segmentácii je našim cieľom rozdeliť obraz do viacerých oblastí, každému pixlu určiť oblasť do ktorej patrí.

3.2 Príznamy

Pri riešení problémov v počítačovom videní sa využívajú príznaky. Príznak v počítačovom videní je merateľný kus dát v obrázku, ktorý je unikátny pre špecifický objekt. Príznak môže reprezentovať napríklad štýl sfarbenia, nejaký tvar, či už čiaru alebo hranu v obraze alebo nejakú časť obrazu. Vďaka dobrému príznaku dokážeme od seba rozlíšiť objekty. Napríklad ak máme rozlíšiť mačku a bicykel tak ako dobrý príznak by mohlo byť, že na obrázku sa nachádza koleso. Hneď by sme vedeli vďaka tomuto príznaku klasifikovať obrázky do týchto dvoch tried. Ak by sme však mali za úlohu zistiť či je na obrázku motorka alebo bicykel, tak by nám tento príznak veľmi nepomohol a museli by sme pozeráť na iné príznaky. Preto zväčša neextrahujeme z obrázku len jeden príznak, ale pre lepšiu detekciu vyberáme viacej príznakov, ktoré tvoria príznakový vektor.

Nie je presná definícia aké príznaky obrázku by sme mali použiť, ale závisí to skôr od nášho cieľu a typu úlohy. Príznaky sa delia na lokálne a globálne. Globálne príznaky sú také, ktoré platia pre celý obrázok. Napríklad ako ako veľmi sú dominantné jednotlivé farby v obrázku. Globálny príznak nám opisuje obraz ako celok a mal by reprezentovať nejakú jeho špecifickú

vlastnosť. Lokálne príznaky sa extrahujú len z určitej zaujímavej oblasti v obrázku, využívajú sa najmä pri objektovej detekcii. Najskôr nájdeme zaujímavé oblasti, ktoré by mohli reprezentovať nejakú zaujímavú vlastnosť alebo nejaký objekt. Následne vytvoríme príznakový vektor pre danú oblasť, ktorý by nám mal poskytnúť zásadnú informáciu o tejto časti obrazu. Treba rátať s tým, že objekt na obrázku môže byť rôznej veľkosti, rôzne natočený, rôzne osvetlený, zašumený, môže sa nachádzať v rôznych častiach obrázku a podobne. Preto naše príznaky by mali byť ideálne invariantné voči týmto zmenám (mali by ich čo najmenej ovplyvňovať). Čím viac invariantné príznaky voči týmto zmenám si zvolíme tým lepšia je naša detekcia.

3.3 Objektová detekcia

Asi najskúmanejším problémom v počítačovom videní je objektová detekcia, ktorá spočíva v rozpoznaní jednotlivých objektov a ich pozícií v digitálnom obraze. K tomuto problému sa dá pristupovať tradičnými metódami počítačového videnia, alebo dnes už veľmi rozšíreným s oveľa lepšími a presnejšími výsledkami, ako pri tradičných metódach a to pomocou hlbokého učenia, ktorých kľúčom je naučiť sa na veľkých dátach extrahovať príznaky tak aby mala detekcia čo najväčšiu presnosť.

3.3.1 Tradičné metódy

Ako prvé vznikli tradičné metódy. Vysvetlíme si ako fungujú, pretože nám to pomôže pochopiť ako funguje dnes najvyužívanejší a najpresnejší prístup hlbokého učenia na ktorý sa zameriame v tejto práci. Tradičné metódy v objektovej detekcii majú zvyčajne tri etapy: vybratie oblasti, extrakcia príznakov, klasifikácia objektu.

V prvej etape sa snažíme lokalizovať objekt. Keďže objekt môže byť rôznej veľkosti, musíme skenovať celý obrázok pomocou posúvného okna rôznej veľkosti. Táto metóda je výpočtovo náročná.

V druhej etape použijeme použijeme metódy ako SIFT [Low99], HOG [DT05] na extrakciu vizuálnych príznakov na rozpoznanie objektu. Tieto príznaky nám poskytujú sémantickú a robustnú reprezentáciu. Avšak kvôli rôznemu osvetleniu, pozadiu a ulhu pohľadu je veľmi náročné manuálne navrhnuť deskriptor príznakov, ktorý by dokonale opísal všetky typy objektov.

V tretej fáze klasifikácie objektu používame zväčša Support Vector Machine(SVM) [CV95] alebo Adaboost [Sch13] pre klasifikáciu cieľových objektov zo všetkých kategórii aby bola reprezentácia viac hierarchická, sémantická a informatívnejšia pre vizuálne rozpoznávanie.

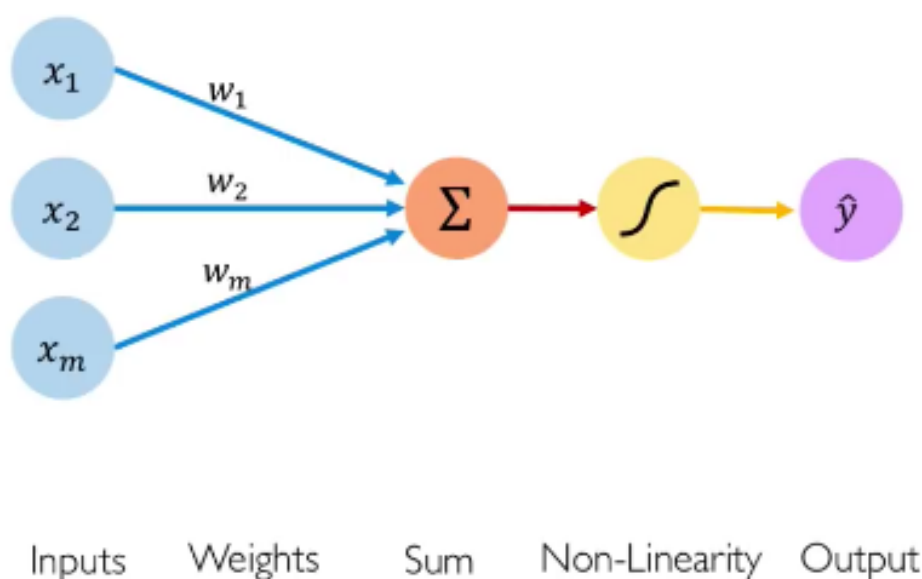
Problémom pri tradičných metódach je výpočtová náročnosť pri generovaní kandidátov na bounding box (obdĺžnik ohraničujúci objekt) pomocou techniky posúvného okna a taktiež manuálne nastavenie extrakcie príznakov nie je vždy veľmi presné. Avšak ich výhodou je, že nepotrebujeme veľký anotovaný dataset a taktiež veľkú výpočtovú silu pri tréningu, ktoré potrebujeme pri prístupe hlbokého učenia.

3.3.2 Metódy hlbokého učenia

Neskôr keď tradičné metódy začali stagnovať, sa začali na riešenie problémov klasifikácie obrázkov, objektovej detekcie a segmentácie využívať metódy hlbokého učenia. Hlavným dôvodom, prečo metódy hlbokého učenia dosahujú lepšie výsledky ako tradičné metódy je, že netreba manuálne voliť príznaky, ale ich úlohou je nájsť najlepšie príznaky pre danú úlohu. Využívajú na to neurónové siete.

Neurónové siete

Základným stavebným blokom neurónovej siete je neurón. Poďme si vysvetliť ako funguje neurónova sieť zložená len z jedného neurónu. Ako vidíme na obrázku 3.1 do neurónu vstupuje $x_1, x_2 \dots x_m$ vstupov. Každý z týchto vstupov má svoju váhu $w_1 \dots w_m$. Najprv každý vstup prenasobíme jeho prislúchajúcou váhou a následne spravíme ich sumu. Ku tejto sume ešte prirátame bias b . Následne výsledok pošleme do nelineárnej aktivačnej funkcie a dostaneme výstup z neurónu. Hlavnou úlohou aktivačnej funkcie je zavedenie nelinearity. Matematicky vyjadrené na obrázku 3.2.



Obr. 3.1: Ukážka jedného neurónu

Neurónová sieť sa skladá zväčša z viacej neurónov a viacerých vrstiev. Výstup neurónu z nižšej vrstvy môže byť vstupom do neurónu vo vyššej vrstve a ako sme si popísali vyššie má svoju váhu, ktorá popisuje silu spojenia

The diagram shows the formula for a neuron's output, $\hat{y} = g \left(w_0 + \sum_{i=1}^m x_i w_i \right)$. Annotations include:

- A purple arrow pointing to \hat{y} labeled "Output".
- A red arrow pointing to the summation term $\sum_{i=1}^m x_i w_i$ labeled "Linear combination of inputs".
- A green arrow pointing to w_0 labeled "Bias".
- A yellow arrow pointing to g labeled "Non-linear activation function".

Obr. 3.2: Vzorec pre neurón

medzi dvoma neurónmi.

Váhy w a bias b sú parametre, ktoré sa pri tréningu neurónovej siete prispôsobujú, tak aby sa minimalizoval rozdiel medzi výstupom siete a očakávaným výstupom.

Najviac využívaným typom neurónových sietí v počítačovom videní sú konvolučné neuronové siete, ktoré sú ideálne na spracovanie dát v mriežkovitom tvare ako napríklad obrázkov. O nich si povieme viac v ďalšej časti.

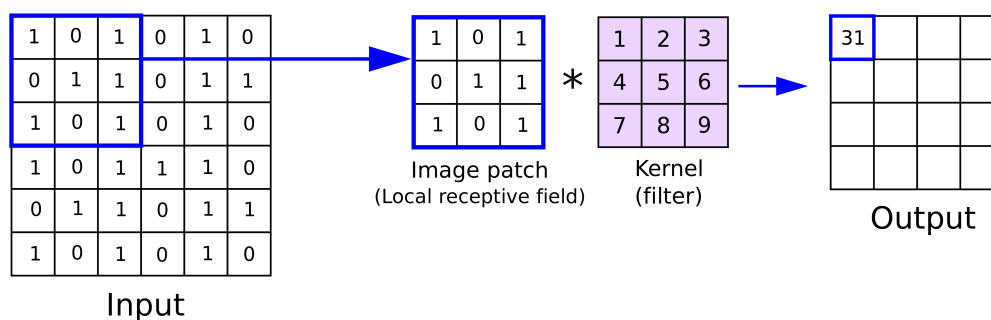
Úvod do CNN

Konvolučné neuronové siete (CNN) [LB⁺95] sú typ neurónovej siete, ktorá obsahuje konvolučné vrstvy. Konvolučné vrstvy skenujú dáta pomocou množiny filtrov, kde každý filter hľadá špecifický vzor v dátach. Vďaka

týmto vrstvám sú CNN ideálne na spracovanie dát mriežkovitého tvaru ako napríklad obrázky. Teraz si vysvetlíme ako fungujú konvolučné vrstvy.

Konvolučné vrstvy

Ako sme si spomenuli v úvode, konvolúcia sa deje pomocou filtrov. Aplikáciu filtra si vysvetlíme pomocou konkrétneho príkladu na obrázku 3.3 kde máme vstupný obrázok veľkosti 6x6 a filter veľkosti 3x3, po aplikácii tohto filtru na obrázok dostaneme výstup veľkosti 4x4. Z obrázku je jasné ako sa vyráta prvá hodnota vo výstupe, pre vyrátanie druhej hodnoty v prvom riadku sa naše modré okno veľkosti 3x3 posunie o 1 doprava. Takýmto spôsobom vyrátame všetky hodnoty v prvom riadku a následne pre ďalší riadok sa posunieme o 1 nadol.



Obr. 3.3: Aplikácia filtra

Toto posunutie sa nazýva stride a dá sa nastaviť aj na vyššie číslo ako 1.

Ďalší nastaviteľný parameter pri konvolúcii je padding kde zväčšíme veľkosť nášho vstupu tak, že pridáme rám okolo nášho vstupu s nulovými hodnotami ako vidíme na obrázku 3.4 bielou farbou pridané hodnoty pri paddingu veľkosti 1. Veľkosť paddingu predstavuje šírku tohto rámu. Padding by nemal byť väčší ako výška alebo šírka filtra. Vďaka paddingu vieme napríklad do-

siahnúť, že rozmery výstupu budú rovnaké ako rozmery vstupu, pri rôznych rozmeroch filtra.

Pri nasvaovaní rozmerov filtra, stridu a paddingu, musí ich kombinácia sedieť na rozmery vstupu. tak aby sa dal prejsť filtrom celý vstup.

Filtrov môže byť pri konvolúcii kludne aj viac ako 1. Viac filtrov nám zvyšuje dimenziu výstupu. Napríklad pri vstupe 6x6 a použití 5 tich filtrov veľkosti 3x3 stride = 1 a padding = 1 dostaneme výstup s rozmermi 5x6x6. V ďalšej časti si predstavíme pooling.

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Obr. 3.4: Znázornenie paddingu pri konvolúcii

Pooling

Po konvolučnej vrstve sa v CNN zvyčajne nachádza poolingová vrstva, ktorá zvyčajne znižuje priestorovú dimenziu. Poznáme niekoľko typov poolingov, vrátane max pooling a average pooling, ale najbežnejším je max pooling, ktorý vezme maximálnu hodnotu každého poolingového regiónu.

Napríklad, ak pooling región je mriežka 2×2 , max pooling vezme najväčšiu zo 4roch hodnôt v mriežke a vráti ju ako jednu hodnotu do výstupnej príznakovej mapy. Toto spôsobí redukciu veľkosti príznakovej mapy a ponecháme len najdôležitejšie príznaky.

Plne prepojené vrstvy

Ďalej po konvolučných a poolingových vrstvách, CNN zvyčajne obsahujú jednu alebo viac plne prepojených vrstiev, ktoré kombinujú príznaky extrahované konvolučnými a poolingovými vrstvami na určenie finálnej predikcie.

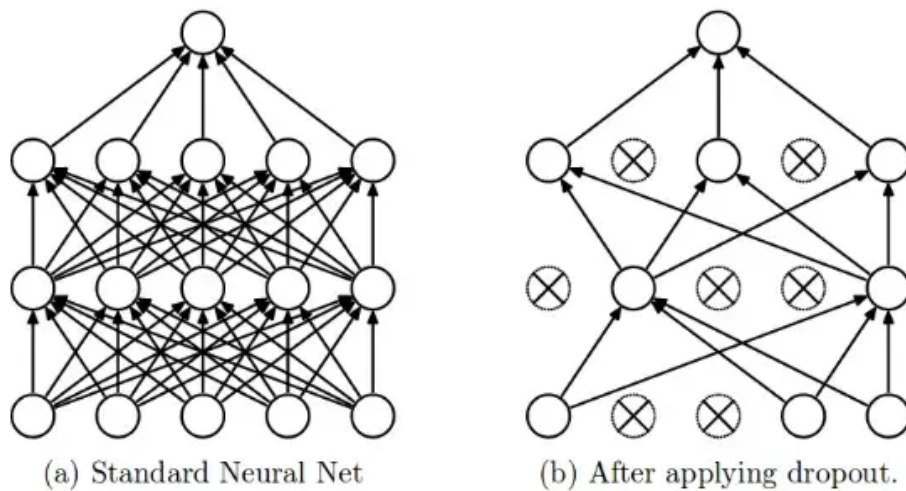
Plne prepojená vrstva je zvyčajne výstupná vrstva, ktorá vracia finálnu predikciu CNN. Počet neurónov vo výstupnej vrstve závisí od tasku, ktorý máme. Napríklad, pre klasifikáciu obrázku, výstupná vrstva môže mať jeden neurón pre každú triedu, a neurón s najvyššou aktivačnou hodnotou by predstavoval predikovanú triedu.

Aktivačné funkcie a regularizácia

Za účelom zavedenia nelinearity do siete zvyčajne CNN obsahujú aktivačné funkcie v konvolučných a plne prepojených vrstvách. Najbežnejšou aktivačnou funkciou je Rectified Linear Unit (ReLU), ktorá má formu $f(x) = \max(0, x)$. Používajú sa aj iné aktivačné funkcie ako napríklad sigmoid a tanh.

Na prevenciu overfittingu a zlepšenie generalizácie, konvolučné neurónové siete využívajú regularizačné techniky ako napríklad dropout.

Dropout náhodne dropne(nepoužije) nejaké percento neurónov v sieti počas každej tréningovej iterácie. ako vidíme na obrázku 3.5



Obr. 3.5: Aplikácia dropoutu

Overfitting je pretrénovanie siete, to znamená, že naša sieť funguje príliš dobre na tréningových dátach, ale má to negatívny vplyv na výkon siete na nových dátach, ktoré neboli videné pri tréningu. Teda celkový výkon našej siete všeobecne na všetkých dátach sa overfittingom znižuje.

Tréning CNN

Tréning CNN zahŕňa prispôbenie váh filtrov a spojení v sieti na minimalizovanie stratovej funkcie, ktorá meria rozdiel medzi predikciou a skutočným labelom. Process trénovanie CNN môže byť rozdelený na nasledovné kroky:

Prvým krokom je vyzbierať anotovaný dataset, ktorý bude použitý na

trénovanie modelu. Tento dataset by mal byť dostatočne veľký a rôznorodý aby sa naša sieť vedela generalizovať na nové obrázky. Pred tréningom siete, je zväčša nevyhnutné predspracovanie dát, aby sme sa uistili, že sú vhodné pre CNN. To môže zahŕňať prispôbenie veľkosti obrázkov alebo augmentáciu dát aplikovaním náhodných transformácií, pre rôznorodosť datasetu a predchádzaniu overfittingu.

Ďalším krokom je rozdelenie datasetu na tréningovú, validačnú a testovaciu množinu. Trénovacia množina je použitá na trénovanie CNN, validačná na vyhodnotenie CNN počas tréningu a testovacia na vyhodnotenie modelu po tréningu. Validačná množina je nápomocná pre nastavenie hyperparametrov CNN. Testovacia množina nám poskytuje približnú schopnosť generalizácie našej siete.

Hyperparametre sú parametre ktoré môžeme upraviť pre optimalizáciu výkonu našej siete. Medzi ne patria napríklad: počet vrstiev v sieti, počet filtrov v konvolučných vrstvách, veľkosť filtrov, stride, padding, pooling, dropout, learning rate, veľkosť mini-batchu, počet epoch.

Learning rate je veľkosť kroku, ktorý spraví optimalizátor na upravenie parametrov siete, optimalizátor si vysvetlíme neskôr.

Veľkosť mini-batchu je počet tréningových príkladov použitých v jednej iterácii tréningu.

Počet epoch je koľko krát prejdeme celým datasetom počas tréningu.

Tretím krokom je návrh CNN a voľba hyperparametrov. Je veľa spôsobov ako navrhnúť CNN, ktoré majú vplyv na jej výkon. Najlepšie je vyskúšať rôzne hyperparametre a architektúry CNN pre nájdenie najlepšej kombinácie k danej úlohe. Dobré je sieť skúšať trénovať a pomaly upravovať.

Ďalším krokom je určenie stratovej funkcie a optimalizačného algoritmu. Stratová funkcia meria ako dobre je náš model schopný predikovať žiadaný

výstup pre konkrétny vstup a jej voľba závisí na type úlohy.

Optimalizačný algoritmus (optimalizátor) je zodpovedný za prispôbovanie parametrov siete na minimalizovanie stratovej funkcie. Najbežnejší optimalizačný algoritmus je stochastic gradient descent (SGD). Iný často používaný optimalizačný algoritmus je napríklad Adam [?].

Gradient Descent (Gradientový zostup) prispôbuje parametre našej siete podľa vzorca na obrázku 3.6. Vypočítaním gradientu stratovej funkcie získame smer k maximu stratovej funkcie v danom bode, teda s našimi aktuálnymi parametrami. My sa snažíme dosiahnuť minimum preto naše parametre upravíme v opačnom smere gradientu vynásobený krokom učenia (learning rate). Tento optimalizačný krok iteratívne opakujeme, kým sa nedostaneme k minimu stratovej funkcie. Gradient descent prispôbuje parametre podľa všetkých tréningových príkladov.

$$X = X - lr * \frac{d}{dX} f(X)$$

Where,

X = *input*

$F(X)$ = *output based on X*

lr = *learning rate*

Obr. 3.6: Gradientový zostup

Stochastic gradient descent sa líši od gradient descentu tým, že neupravuje parametre vzhľadom na všetky tréningové príklady, ale len vzhľadom

na niekoľko tréningových príkladov, počet týchto tréningových príkladov závisí od veľkosti mini-batchu. SGD je rýchlejšie a menej výpočtovo náročné, taktiež sa pridáva šum do optimalizačného procesu, čo znižuje šancu, že algoritmus skončí v lokálnom minime.

Po trénovaní CNN je dôležité vyhodnotiť výkon na testovacej množine. Testovacia množina by mala byť dostatočne veľká aby nám ponkla spoľahlivé vyhodnotenie nášho modelu a nemala by byť použitá pri tréningu.

Výkon CNN môže byť vyhodnotený pomocou rôznych metrík. Zavisí od úlohy, ktorú metriku je pre nás vhodné použiť.

3.4 Few-shot object detection(FSOD)

Problémom pri väčšine algoritmov objektovej detekcie je, že vyžadujú veľký dataset anotovaných obrázkov na tréning modelu, čo môže byť drahé a časovo náročné.

Few-shot object detection je varianta objektovej detekcie, ktorá sa snaží učiť z malého datasetu. Je to inšpirované few-shot learningom, čo je typ strojového učenia, ktorý sa učí na malých tréningových dátach. Few-shot learning si získal v posledných rokoch veľa pozornosti, vďaka jeho schopnosti adaptovať sa novým taskom s malým množstvom dát, čo je dôležité v prípade, že nemáme dostatok anotovaných dát.

Few-shot object detection je náročný problém, pretože model sa musí naučiť charakteristiky objektov z malého množstva príkladov, čo je náročné vzhľadom na komplexnosť a rôznorodosť objektov. Navyše model musí rozpoznať nové triedy, ktoré neboli videné počas tréningu, čo vyžaduje dobré rozlišovanie medzi odlišnými triedami.

Napriek náročnosti, je to dôležitý problém, pretože má potenciál výrazne

znižiť počet anotovaných dát potrebných na objektovú detekciu.

3.4.1 Prístupy k FSOD

Sú viaceré prístupy, ktoré boli navrhnuté na FSOD, môžu byť zhrnuté do troch hlavných kategórií: meta-learning, transfer learning a augmentácia dát.

Meta-learning

Meta-learning je prístup, ktorý sa snaží naučiť sa učiť. Sústreďí sa na tréning modelu, ktorý sa vie rýchlo prispôbiť novým úlohám iba vďaka veľmi malému počtu obrázkov. Tento prístup zvyčajne zahŕňa tréning modelu, ktorý sa vie učiť z malého počtu dát buď použitím vonkajšej pamäti alebo optimalizačného algoritmu. Napríklad Model-Agnostic Meta-Learning (MAML) [FAL17] algoritmus používa gradientový optimalizačný algoritmus na tréning modelu, ktorý sa vie prispôbiť novým úlohám malým počtom aktualizácií gradientu. Algoritmus MAML bol aplikovaný na few-shot object detection pri fine-tuningu predtrénovaného modelu na objektovú detekciu na malom počte obrázkov.

Transfer learning

Transfer learning je technika pri ktorej sa využívajú parametre natrénovanej siete z jednej úlohy ako iníciaálne parametre pre sieť na novú veľmi podobnú úlohu. Napríklad sieť trénovaná na veľkom datasete obrázkov zvierat by mohla byť použitá ako iníciaálna sieť pre tréning siete na rozpoznávanie špecifického typu zvierat ako napríklad plemená psov. Použitím siete s predtrenovanými váhami, sa naša sieť naučí rýchlejšie rozpoznávať plemená psov a stačí na to menej dát.

Augmentácia dát

Prístup augmentácie dát spočíva v rozmnožení malého množstva dát pomocou aplikovania rôznych transformácií. Zvýšením počtu dát, sa model môže naučiť robustnejšie príznaky. Pri augmentácii sa používajú transformácie ako rotácia, škálovanie, zašumenie. Používame transformácie obrazu, ktoré menia obraz, ale nemenia jeho sémantický obsah.

3.4.2 Datasets pre vyhodnotenie FSOD

Na vyhodnotenie výkonu FSOD algoritmov, výskumníci používajú verejne dostupné datasety a vyhodnocovacie metriky. Tieto datasety a metriky slúžia na porovnanie výkonu odlišných algoritmov a ich všeobecnosti. Najpoužívanejšie datasety sú:

COCO dataset [?], ktorý obsahuje 80 tried a viac ako 330 000 anotovaných obrázkov. VOC dataset [?], je to populárny dataset, ktorý obsahuje 20 tried a viac ako 11 000 obrázkov.

Tieto datasety sú používané ako štandard pre few-shot object detection, vyberie sa z nich niekoľko tried, ktoré sa považujú ako novel classes (triedy s malým počtom anotovaných dát), pre tieto triedy sa použije iba zopár anotovaných obrázkov(few-shot) a zvyšné triedy sa použijú na predtrénovanie modelu.

3.4.3 Aktuálne riešenia FSOD

K FSOD bolo publikovaných viacero článkov, a veľa autorov použilo iné datasety a spôsoby vyhodnotenia ich modely. Preto je náročné ich porovnanie. Avšak, vo všeobecnosti meta-learningové prístupy zvyknú dosahovať lepšie výsledky ako transfer learning alebo prístup augmentácie dát. Hlavne

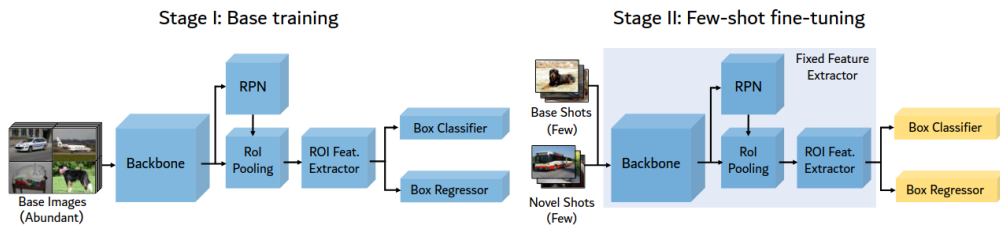
preto, že meta-learningové prístupy sú špeciálne navrhnuté učiť sa z malého počtu príkladov.

Avšak, je potrebné zmieniť, že výkon few-shot object detection modelu veľmi závisí od konkrétnej implementácie, zvolených dát a zvolených metrík. A taktiež treba brať do úvahy výpočtovú a pamäťovú náročnosť.

Frustratingly simple few-shot object detection

Frustratingly simple few-shot object detection [?] je metóda, ktorú sme sa rozhodli použiť v tejto práci. Kľúčová myšlienka za touto metódou je naučiť sa detekovať objekty tréningom na množine základných tried (base classes) s veľkým počtom anotovaných obrázkov a následne spraviť fine-tuning detektora na malom množstve anotovaných obrázkov z nových tried (novel classes).

Ako vidíme na obrázku 3.7 model sa delí na dve časti: feature extractor (v pravej časti obrázku označený modrou farbou) a box predictor (v pravej časti obrázku označený žltou farbou).



Obr. 3.7: Model pre frustratingly simple few shot object detector

Tréning tohto modelu sa delí na 2 etapy: V prvej etape sa vykoná base tréning na base classes na to sa využíva Fster-R-CNN [?], natrénuje sa feature extractor aj box predictor pomocou stratovej funkcie:

kde L_{rpn} sa aplikuje na výstup z RPN na rozlíšenie popredia od pozadia, L_{cls} je cross-entropy loss pre box classifier a L_{loc} je stratová funkcia pre

$$\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}$$

Obr. 3.8: Stratová funkcia FSFSODT

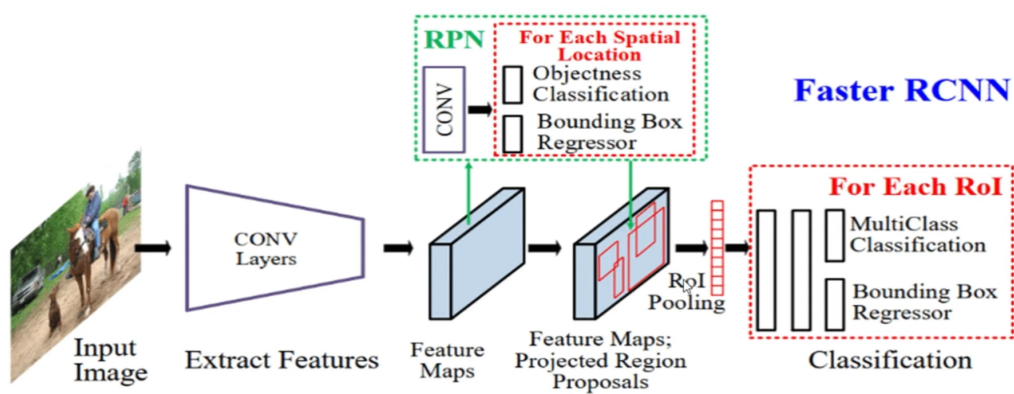
box regressor. Box regressor predikuje pozíciu bounding boxu, box classifier klasifikuje objekt do tried.

V druhej etape (few-shot fine-tuning) vytvoríme pre každú z tried (novel a base) malý tréningový set. Pre novel classes priradíme náhodné inicializované váhy do siete pre box predictor. A následne robíme finetuning ale len na poslednej vrstve nášho modelu, celý feature extraktor ostáva zachovaný a trénujeme len box classifier a box regressor. Použijeme rovnakú loss funkciu a 20x nižší learning rate.

Kľúčovým prvkom tejto metódy je oddelenie učenia sa reprezentácii príznakov a učenia sa predikovania boxov. Keďže príznaky, ktoré sme sa naučili používať na base classes môžeme využiť pre novel classes.

Faster R-CNN

Ako vidíme na obrázku 3.9 Faster-RCNN najprv extrahuje príznaky zo vstupného obrázku, vytvorí príznakovú mapu. Následne sa táto príznaková mapa použije ako vstup pre region proposal network (RPN), ktorá má za úlohu predikovať regióny v ktorých by sa mal nachádzať objekt. Následne sa tieto regióny pridajú do príznakových máp, ktoré sme dostali z obrázku pri prvom kroku. Potom vezmeme každý tento región zvlášť a pomocou plne prepojených vrstiev sa klasifikuje.



Obr. 3.9: Faster R-CNN

Kapitola 4

Predchádzajúce riešenia

Kapitola 5

Návrh modelu

Kapitola 6

Implementácia

Kapitola 7

Výskum a výsledky

V tejto kapitole sa budeme venovať metódam a postupom vykonávania výskumu pre našu diplomovú prácu, vďaka ktorým sme sa dopracovali k finálnemu riešeniu našej práce.

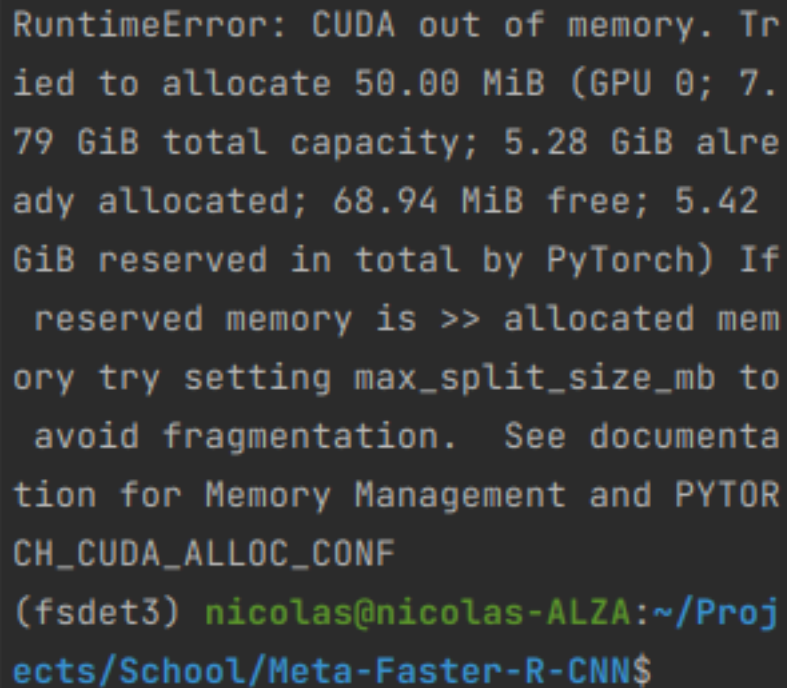
Ako prvé preskúmame few-shot object detection algoritmy a vyskúšame ako fungujú v praxi a ako ich môžeme využiť v našej práci na učenie nových objektov a gest vďaka malému počtu anotovaných obrázkov.

Preskúmal som dva prístupy: Frustratingly simple few shot object detection [WHD⁺20], Meta faster r-cnn: Towards accurate few-shot object detection with attentive feature alignment [HHM⁺22]

Pri testovaní druhého spomínaného prístupu som zistil, že je príliš pamäťovo náročný a nebol som schopný ani po stiahnutí predtrénovaných modelov po prvých dvoch krokoch: meta-training, training base classes detection head, vykonať posledný pre mňa najôležitejší krok ktorý závisel na nových few-shot triedach (novel classes) fine-tuning. Moja grafická karta nemala dostatočnú pamäť 8GB a skúšal som to aj na virtual machine s grafickou kartou nvidia T4 16GB.

Autori trénovali model na 4 20GB GPU. A ja som dostal nasledovný

výstup aj po úprave trénovacích parametrov ako batch size, learning rate, tak aby stačilo čo najmenej pamäte:

A screenshot of a terminal window with a dark background and light-colored text. The text shows a Python RuntimeError: CUDA out of memory. It provides details about GPU 0's memory usage: 50.00 MiB allocated, 7.79 GiB total capacity, 5.28 GiB already allocated, 68.94 MiB free, and 5.42 GiB reserved. It suggests setting max_split_size_mb to avoid fragmentation and points to PyTorch documentation. The prompt is (fsdet3) nicolas@nicolas-ALZA:~/Projects/School/Meta-Faster-R-CNN\$.

```
RuntimeError: CUDA out of memory. Tr
ied to allocate 50.00 MiB (GPU 0; 7.
79 GiB total capacity; 5.28 GiB alre
ady allocated; 68.94 MiB free; 5.42
GiB reserved in total by PyTorch) If
reserved memory is >> allocated mem
ory try setting max_split_size_mb to
avoid fragmentation. See documenta
tion for Memory Management and PYTOR
CH_CUDA_ALLOC_CONF
(fsdet3) nicolas@nicolas-ALZA:~/Proj
ects/School/Meta-Faster-R-CNN$
```

Obr. 7.1: Výstup pri fine-tuningu pomocou Meta Faster-RCNN

Avšak Frustratingly simple few-shot object detection prístup sa mi po-darilo vyskúšať a otestovať, a po prispôsobení trénovacích parametrov k mo-jej obmedzenej pamäťovej kapacite som bol schopný spustiť všetky fázy tré-ningu.

7.1 Testovanie Frustratingly simple few-shot object detection

Dataset:

Pri few-shot object detection, potrebujeme mať dataset rozdelený na triedy ku ktorým máme veľké množstvo anotovaných dát(base classes), a na triedy ku ktorým máme malé množstvo anotovaných dát(novel classes).

Ako dataset som použil pascal voc, ako base classes som použil triedy: aeroplane, bicycle, boat, bottle, car, cat, chair, diningtable, dog, horse, person, pottedplant, sheep, train, tvmonitor.

Ako novel classes som použil triedy z pascal voc: bird, bus, cow, motorbike, sofa a taktiež som pridal jednu vlastnu triedu apple(kde som anotoval 85 obrázkov pomocou labellmg), na trening(v druhej fáze pri finetuningu) bolo použitých len 5 obrázkov z každej triedy, ostatné boli použité na testovanie

Pre každú triedu som vytvoril 5-shot split, vybral som 5 obrázkov pre každú triedu pre fine-tuning.

Tréning:

Najprv som si stiahol predtrénovaný base model, ktorý bol trénovaný čisto iba na 15 base classes. Následne som náhodne inicializoval váhy pre novel classes v poslednej vrstve modelu. A potom som spustil finetuning poslednej vrstvy modelu.

Pri fine-tuningu som použil tieto parametre:

Vyhodnotenie:

Ako vidíme hodnoty pre base classes nám ostali vysoké, a pri novel classes sa napriek tomu, že sme mali na tréning len 5 vzoriek a testovacia množina

```
_BASE_: "../../../Base-RCNN-FPN.yaml"
MODEL:
  WEIGHTS: "checkpoints/voc/faster_rcnn/faster_rcnn_R_101_FPN_base1/model_reset_surgery.pth"
  MASK_ON: False
  RESNETS:
    DEPTH: 101
  ROI_HEADS:
    NUM_CLASSES: 21
    OUTPUT_LAYER: "CosineSimOutputLayers"
    FREEZE_FEAT: True
  BACKBONE:
    FREEZE: True
  PROPOSAL_GENERATOR:
    FREEZE: True
INPUT:
  MIN_SIZE_TRAIN: (480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800)
  MIN_SIZE_TEST: 800
DATASETS:
  TRAIN: ('voc_2007_trainval_all1_5shot',)
  TEST: ('voc_2007_test_all1',)
SOLVER:
  IMS_PER_BATCH: 2
  BASE_LR: 0.000125
  STEPS: (144000,)
  MAX_ITER: 160000
  CHECKPOINT_PERIOD: 1000
  WARMUP_ITERS: 10
  OUTPUT_DIR: "checkpoints/voc/faster_rcnn/faster_rcnn_R_101_FPN_ft_normalized_all1_5shot_randnovel"
```

Obr. 7.2: Tréningové parametre pri fine-tuningu

bola veľmi variabilná tak sme dostali mAP50 u každej novel class nad 30.

Avšak myslel som si, že fine-tuning bude prebiehať veľmi rýchlo, keďže prebieha len na zopár obrázkoch z každej triedy a robot sa bude môcť učiť v real-time nové objekty, ale len samotný fine-tuning pri 5-shot trval na mojom GPU 5 hodín. Takže na učenie objektov v real-time sa to nebude dať použiť. Ale bude sa to dať použiť na učenie nových gest aj objektov, len pomocou zopár anotácií.

```
[12/04 20:27:50 fsdet.evaluation.pascal_voc_evaluation]: Evaluate per-class mAP50:
```

aeroplane	bicycle	boat	bottle	car	cat	chair	diningtable
87.234	85.554	65.447	72.969	87.811	87.599	53.277	67.249

dog	horse	person	pottedplant	sheep	train	tvmonitor	bird
84.006	84.268	85.104	51.820	81.892	87.834	82.011	31.365

bus	cow	motorbike	sofa	apple
68.553	58.339	60.000	37.702	35.056

```
[12/04 20:27:50 fsdet.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
```

AP	AP50	AP75	bAP	bAP50	bAP75	nAP	nAP50	nAP75
44.381	69.290	49.104	49.950	77.605	55.450	30.459	48.502	33.239

Obr. 7.3: Výsledky tréningu

Kapitola 8

Záver

Literatúra

- [CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.
- [EM21] Lorenzo Natale Elisa Maiettini, Vadim Tikhanoff. Weakly-supervised object detection learning through human-robot interaction. *IEEE*, (1):8, 2021.
- [FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
- [HHM⁺22] Guangxing Han, Shiyuan Huang, Jiawei Ma, Yicheng He, and Shih-Fu Chang. Meta faster r-cnn: Towards accurate few-shot object detection with attentive feature alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 780–789, 2022.

- [LB⁺95] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [Low99] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
- [Sch13] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.
- [VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.
- [WHD⁺20] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020.

Zoznam obrázkov

3.1	Úkážka jedného neurónu	7
3.2	Vzorec pre neurón	8
3.3	Aplikácia filtra	9
3.4	Znázornenie paddingu pri konvolúcii	10
3.5	Aplikácia dropoutu	12
3.6	Gradientový zostup	14
3.7	Model pre frustratingly simple few shot object detector . . .	18
3.8	Stratová funkcia FSFSODT	19
3.9	Faster R-CNN	19
7.1	Výstup pri fine-tuningu pomocou Meta Faster-RCNN	25
7.2	Tréningové parametre pri fine-tuningu	27
7.3	Výsledky tréningu	28